STIC Search Report
EIC 2100

USPTO

STIC Database Tracking Number: 119185

TO: Fred Ehichoya
Location: Cpk2 4D43
Art Unit : 2172
Friday, April 09, 2004

Case Serial Number: 09/826710

From: David Holloway
Location: EIC 2100
PK2-4B30
Phone: 308-7794

david.holloway@uspto.gov

Search Notes

Dear Examiner Ehichoya,

Attached please find your search results for above-referenced case.
Please contact me if you have any questions or would like a re-focused search.

David

Scientific ... Information Center
STIC
USPTO

STIC EIC [illegible]100
Search Request Form 119185

Today's Date: 4/9/08	What date would you like to use to limit the search? Priority Date: ~~4/9/08~~ 4/5/01 Other:

Name FRED EHICHIOYA	Format for Search Results (Circle One): PAPER DISK EMAIL
AU 2172 Examiner # 79719	Where have you searched so far?
Room # 4D43 Phone 305-8039	USP DWPI EPO JPO ACM IBM TDB
Serial # 09826710	IEEE INSPEC SPI Other ____

Is this a "Fast & Focused" Search Request? (Circle One) YES NO
A "Fast & Focused" Search is completed in 2-3 hours (maximum). The search must be on a very specific topic and meet certain criteria. The criteria are posted in EIC2100 and on the EIC2100 NPL Web Page at http://ptoweb/patents/stic/stic-tc2100.htm.

What is the topic, novelty, motivation, utility, or other specific details defining the desired focus of this search? Please include the concepts, synonyms, keywords, acronyms, definitions, strategies, and anything else that helps to describe the topic. Please attach a copy of the abstract, background, brief summary, pertinent claims and any citations of relevant art you have found.

See claim 1 particularly "identifying a minimal portion of said data ...."
See page 6 of attached for definition of "fragments"
See pages 8 and 9 for explaination of minimal portion.

STIC Searcher David Holloway	Phone 308-7794
Date picked up 4-9-04	Date Completed 4-9-04

sira
Search and Information Resources Administration

```
Set     Items   Description
S1    6132121   PARTIAL? OR FUZZY? OR PORTION? OR SIGNIFICANT? OR PORTION?
             OR FRACTION? OR FRAGMENT?
S2    8915834   MATCH? OR QUER? OR SEARCH? OR RETRIEV? OR LOCAT? OR IDENTI-
             F?
S3   10406846   STRING? OR SEARCHSTRING? OR CHARACTER? OR ALPHANUMERIC? OR
             LETTER? OR WORD? OR TERM? OR PHRASE?
S4      28275   (SINGLE OR ONE OR INDIVIDUAL? OR UNIQUE?)(N)(OCCUR? OR APP-
             EAR? OR MATCH?)
S5     416469   S2(N)(ENGINE? OR SOFTWARE? OR APPLICATION? OR SYSTEM? OR P-
             ROGRAM? OR CRAWLER? OR IA OR BOT OR ROBOT OR AGENT? OR TOOL?)-
             OR SEARCHENGINE?
S6        150   S1(S)S2(S)S3(S)S4
S7       1702   S1(2N)S2(S)S5
S8        273   S1(2N)S3(S)S2(S)S5
S9          3   S6(S)(S7 OR S8)
S10         3   RD (unique items)
S11        10   S6(S)S5
S12         4   S7(S)S4
S13         2   S8(S)S4
S14        12   S10 OR S11 OR S12 OR S13
S15        11   RD (unique items)
S16         7   S15 NOT PD=20010405:20040409
File 275:Gale Group Computer DB(TM) 1983-2004/Apr 09
         (c) 2004 The Gale Group
File  47:Gale Group Magazine DB(TM) 1959-2004/Apr 09
         (c) 2004 The Gale group
File 636:Gale Group Newsletter DB(TM) 1987-2004/Apr 09
         (c) 2004 The Gale Group
File  16:Gale Group PROMT(R) 1990-2004/Apr 09
         (c) 2004 The Gale Group
File 624:McGraw-Hill Publications 1985-2004/Apr 08
         (c) 2004 McGraw-Hill Co. Inc
File 484:Periodical Abs Plustext 1986-2004/Apr W1
         (c) 2004 ProQuest
File 613:PR Newswire 1999-2004/Apr 09
         (c) 2004 PR Newswire Association Inc
File 813:PR Newswire 1987-1999/Apr 30
         (c) 1999 PR Newswire Association Inc
File 696:DIALOG Telecom. Newsletters 1995-2004/Apr 08
         (c) 2004 The Dialog Corp.
File 621:Gale Group New Prod.Annou.(R) 1985-2004/Apr 09
         (c) 2004 The Gale Group
File 674:Computer News Fulltext 1989-2004/Apr W1
         (c) 2004 IDG Communications
File 369:New Scientist 1994-2004/Apr W1
         (c) 2004 Reed Business Information Ltd.
File 160:Gale Group PROMT(R) 1972-1989
         (c) 1999 The Gale Group
File  15:ABI/Inform(R) 1971-2004/Apr 09
         (c) 2004 ProQuest Info&Learning
File  13:BAMP 2004/Mar W3
         (c) 2004  The Gale Group
File 647:CMP  Computer Fulltext 1988-2004/Mar W4
         (c) 2004 CMP Media, LLC
File 148:Gale Group Trade & Industry DB 1976-2004/Apr 09
         (c)2004 The Gale Group
```

16/3,K/1 (Item 1 from file: 47)
DIALOG(R)File 47:Gale Group Magazine DB(TM)

05862119 SUPPLIER NUMBER: 63842650 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Text Retrieval Products for Libraries.(Technology Information)(Statistical Data Included)
Saffady, William
Library Technology Reports, 36, 2, 5
March, 2000
DOCUMENT TYPE: Statistical Data Included ISSN: 0024-2586
LANGUAGE: English RECORD TYPE: Fulltext
WORD COUNT: 35970 LINE COUNT: 03217

... for the ZyFIND component's excellent repertoire of information retrieval capabilities. Few, if any text **retrieval** **programs** , can **match** ZyIMAGE's ability to import **word** processing files, spreadsheets, and databases created by popular and all-but-forgotten programs. ZyFIND combines familiar components, such as **phrase** **searching** and Boolean operators, with such unusual features as quorum **searching** and the application of relational expressions to numbers embedded in text files. Other notable **retrieval** capabilities include versatile proximity commands, root **word** **searching** , suffix **matching** , **single** and multiple wildcard **characters** , **fuzzy** **searching** , and a preconfigured thesaurus for synonym selection.

Most program operations, including indexing and searching, are...

T6/3,K/4 (Item 1 from file: 15)
DIALOG(R)File 15:ABI/Inform(R)

02510113 258853721
Project delivery system selection: A case-based reasoning framework
Ribeiro, Francisco Loforte
Logistics Information Management v14n5/6 PP: 367-375 2001
ISSN: 0957-6053 JRNL CODE: LIM
WORD COUNT: 4951

...TEXT: score is calculated and the highest ranking cases are then presented to the user.

The **system searches** the project delivery system cases using the hierarchical search algorithm, first looking for cases exactly matching the specified new case problem, and then for **partial matches** . Exactly **matching** are those whose are the same as those specified for a new case problem. **Partial matches** , in order of preference, are project delivery system cases **matching one** or two, or the three indices fully or partially. Given a description of the new...

Set	Items	Description
S1	1845078	PARTIAL? OR FUZZY? OR PORTION? OR SIGNIFICANT? OR PORTION? OR FRACTION? OR FRAGMENT?
S2	1510430	MATCH? OR QUER? OR SEARCH? OR RETRIEV? OR LOCAT? OR IDENTI-F?
S3	2435468	STRING? OR SEARCHSTRING? OR CHARACTER? OR ALPHANUMERIC? OR LETTER? OR WORD? OR TERM? OR PHRASE?
S4	2250	(SINGLE OR ONE OR INDIVIDUAL? OR UNIQUE?)(N)(OCCUR? OR APP-EAR? OR MATCH?)
S5	69	S1 AND S2 AND S3 AND S4
S6	385	S2 AND S4 AND S3
S7	4347	S1(2N)S2 AND S3
S8	24	S5 AND IC=G06F?
S9	19	S8 NOT AD>20010405
S10	10	S6 AND S7
S11	26	S10 OR S9
S12	22	S11 NOT AD>20010405
S13	22	IDPAT (sorted in duplicate/non-duplicate order)
S14	22	IDPAT (primary/non-duplicate records only)

File 347:JAPIO Nov 1976-2003/Dec(Updated 040402)

File 350:Derwent WPIX 1963-2004/UD,UM &UP=200419

14/5/1 (Item 1 from file: 350)
DIALOG(R)File 350:Derwent WPIX

015833463 **Image available**
WPI Acc No: 2003-895667/200382
XRPX Acc No: N03-714602

Key phrase **producing method for multimedia applications, involves processing feature vectors generated for each frames, and applying predetermined rules to marked vectors in order to select label as key** phrase **of song**

Patent Assignee: HEWLETT-PACKARD DEV CO LP (HEWP)
Inventor: CHU S M; LOGAN B T
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6633845	B1	20031014	US 2000545893	A	20000407	200382	B

Priority Applications (No Type Date): US 2000545893 A 20000407
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6633845	B1		16	G10L-015/28	

Abstract (Basic): US 6633845 B1

NOVELTY - The method involves dividing a part of a song into a set of frames, and generating a feature vector for each frame. The feature vectors are processed to **identify** songs structure. The vectors related with different structural parts of the song having different labels are marked. Predetermined rules are applied to the marked vectors for selecting **single occurrence** of a chosen label as a key **phrase** (214) of the song.

DETAILED DESCRIPTION - Each feature vector has parameters whose values are **characteristics** of that **portion** of the song contained within the respective frame. INDEPENDENT CLAIMS are also included for the following:

(a) a system to produce a key **phrase** for a song

(b) a computer readable medium to produce a key **phrase** for a song.

USE - Used for producing key **phrase** in multimedia applications, databases and **search** engines.

ADVANTAGE - The method automatically generates the key **phrase** or summary of a song. The method employs the summary as an index to the song so that the user can **identify** the song by hearing the key **phrases** .

DESCRIPTION OF DRAWING(S) - The drawing shows a block diagram of a song summarization system.

Signal processor (202)
Vector extraction engine (204)
Key **phrase identifier** logic (208)
Audio input (210)
Key **phrase** (214)
pp; 16 DwgNo 2/7

Title Terms: KEY; **PHRASE** ; PRODUCE; METHOD; APPLY; PROCESS; FEATURE; VECTOR; GENERATE; FRAME; APPLY; PREDETERMINED; RULE; MARK; VECTOR; ORDER; SELECT; LABEL; KEY; **PHRASE** ; SING
Derwent Class: P75; P86; T01; W04
International Patent Class (Main): G10L-015/28
International Patent Class (Additional): B41J-003/34; **G06F-007/00** ; G10G-007/00; G10L-021/06
File Segment: EPI; EngPI

14/5/3 (Item 3 from file: 350)
DIALOG(R)File 350:Derwent WPIX

013979168 **Image available**
WPI Acc No: 2001-463382/200150
XRPX Acc No: N01-343477

Computer readable medium for word **processing system, has condensed lexion database with data tree having nodes containing reading pair** identification **number and instructions for mapping reading pair ID number array**

Patent Assignee: MICROSOFT CORP (MICR-N)
Inventor: CAI P P; HALSTEAD P H
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6175834	B1	20010116	US 98104257	A	19980624	200150	B

Priority Applications (No Type Date): US 98104257 A 19980624
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6175834	B1		21	G06F-017/00	

Abstract (Basic): US 6175834 B1

NOVELTY - The condensed lexion database (CLD) has data tree having nodes, each including reading pair ID numbers (RID) and computer executable instructions for mapping RID array onto CLD. Reading pair database (RPD) is accessed to **match one** reading unit in selected **word** , to either of reading units of reading pairs in RPD and **matching** RID is **retrieved** . Each **word** is reformed as RID array which is mapped onto the CLD.

DETAILED DESCRIPTION - The medium has reading pair database (RPD) having several reading pairs and several reading pair **identification** numbers (RIDs). Each of the reading pairs have two reading units in two writing system respectively. Each of the RIDs correspond to one of the reading pairs. The RPD is accessed to **match one** reading unit of the **word** with reading units in RPD. A reply message is output to indicate whether mapping of RID array onto CLD is successful or unsuccessful.

INDEPENDENT CLAIMS are also included for the following:

(a) Consistency checking method;

(b) Common spelling variants generating method;

(c) Reading pair database generating method

USE - In **identification** of inconsistently spelled Japanese **words** in document.

ADVANTAGE - All acceptable spelling variants of particular Japanese **word** is **identified** and generated substantially. Spelling variants that are used inconsistently with other spelling variants in the same document are **identified** . The statistics of spelling variant uses is maintained within particular document which enables consistency checker to **identify** lesser used variants.

DESCRIPTION OF DRAWING(S) - The figure shows the pictorial representation of **portions** of CLD.

pp; 21 DwgNo 6/8

Title Terms: COMPUTER; READ; MEDIUM; **WORD** ; PROCESS; SYSTEM; CONDENSATION; DATABASE; DATA; TREE; NODE; CONTAIN; READ; PAIR; **IDENTIFY** ; NUMBER; INSTRUCTION; MAP; READ; PAIR; ID; NUMBER; ARRAY
Derwent Class: T01
International Patent Class (Main): **G06F-017/00**
File Segment: EPI

14/5/6 (Item 6 from file: 350)
DIALOG(R)File 350:Derwent WPIX

011638359 **Image available**
WPI Acc No: 1998-055267/199806
XRPX Acc No: N98-043771

Method of facilitating access to selectable element on graphical user interface - involves matching one **or more** characters **received from** character **based input device with** character portion **of at least one selectable element within multiplicity of lexically unordered selectable elements**

Patent Assignee: SUN MICROSYSTEMS INC (SUNM)
Inventor: GENTNER D R; JOHNSON E; NIELSEN J
Number of Countries: 025 Number of Patents: 004
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
EP 816990	A2	19980107	EP 97304491	A	19970625	199806	B
JP 10116294	A	19980506	JP 97184597	A	19970626	199828	
US 5884318	A	19990316	US 96670952	A	19960626	199918	
US 5963950	A	19991005	US 96670952	A	19960626	199948	

Priority Applications (No Type Date): US 96670952 A 19960626
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
EP 816990	A2	E	22	G06F-003/023	
Designated States (Regional): AL AT BE CH DE DK ES FI FR GB GR IE IT LI LT LU LV MC NL PT RO SE SI					
JP 10116294	A		29	G06F-017/30	
US 5884318	A			G06F-017/30	
US 5963950	A			G06F-017/30	

Abstract (Basic): EP 816990 A

The method involves receiving one or more **characters** from a **character** based input device. The one or more **characters** received from the **character** based input device are compared with the **character portion** from one or more of the multiplicity of lexically unordered selectable elements. The one or more **characters** received from the **character** based input device are **matched** with the **character portion** of at least one selectable element within the multiplicity of lexically unordered selectable elements.

A selectable element which **matched** the one or more **characters** received from the **character** based input device is armed. A previously armed selectable element is disarmed before arming the selectable element which **matched** the one or more **characters** received from the **character** based input device. The armed selectable element is selected in response to receiving an actuation input signal which indicates the armed selectable element should be selected.

ADVANTAGE - Allows user to quickly **search** and select a selectable element by typing minimum number of **character** .

Dwg.5/11

Title Terms: METHOD; FACILITATE; ACCESS; SELECT; ELEMENT; GRAPHICAL; USER; INTERFACE; **MATCH** ; ONE; MORE; **CHARACTER** ; RECEIVE; **CHARACTER** ; BASED; INPUT; DEVICE; **CHARACTER** ; **PORTION** ; ONE; SELECT; ELEMENT; MULTIPLICITY ; SELECT; ELEMENT
Derwent Class: T01
International Patent Class (Main): **G06F-003/023** ; **G06F-017/30**
International Patent Class (Additional): **G06F-003/14**
File Segment: EPI

```
Set     Items   Description
S1     1845078   PARTIAL? OR FUZZY? OR PORTION? OR SIGNIFICANT? OR PORTION?
                  OR FRACTION? OR FRAGMENT?
S2     1510430   MATCH? OR QUER? OR SEARCH? OR RETRIEV? OR LOCAT? OR IDENTI-
                  F?
S3     2435468   STRING? OR SEARCHSTRING? OR CHARACTER? OR ALPHANUMERIC? OR
                  LETTER? OR WORD? OR TERM? OR PHRASE?
S4        2250   (SINGLE OR ONE OR INDIVIDUAL? OR UNIQUE?)(N)(OCCUR? OR APP-
                  EAR? OR MATCH?)
S5          69   S1 AND S2 AND S3 AND S4
S6         385   S2 AND S4 AND S3
S7        4347   S1(2N)S2 AND S3
S8          24   S5 AND IC=G06F?
S9          19   S8 NOT AD>20010405
S10         10   S6 AND S7
S11         26   S10 OR S9
S12         22   S11 NOT AD>20010405
S13         22   IDPAT (sorted in duplicate/non-duplicate order)
S14         22   IDPAT (primary/non-duplicate records only)
S15      34977   S2(N)(ENGINE? OR SOFTWARE? OR APPLICATION? OR SYSTEM? OR P-
                  ROGRAM? OR CRAWLER? OR IA OR BOT OR ROBOT OR AGENT? OR TOOL?)
                  OR SEARCHENGINE?
S16         51   S15 AND (S4 OR (MINIMUM OR MINIMAL)()S1)
S17         27   S16 AND IC=(G06F? OR H04L?)
S18         24   S17 NOT S11
S19         13   S18 NOT AD>20010405
File 347:JAPIO Nov 1976-2003/Dec(Updated 040402)
         (c) 2004 JPO & JAPIO
File 350:Derwent WPIX 1963-2004/UD,UM &UP=200419
         (c) 2004  Thomson Derwent
```

19/5/7 (Item 7 from file: 350)
DIALOG(R)File 350:Derwent WPIX

013455712 **Image available**
WPI Acc No: 2000-627655/200060
XRPX Acc No: N00-465000

Information retrieval system **using natural language queries in Internet, analyzes language based database and natural language query to generate database keywords and query keywords, respectively**

Patent Assignee: NOVELL INC (NOVE-N)
Inventor: AKKER D V D; DE BIE P; DE HITA C R; DEUN K V; GOVAERS E C E; LAVIOLETTE S; MACPHERSON M; PLATTEAU F M J
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6081774	A	20000627	US 97916628	A	19970822	200060	B

Priority Applications (No Type Date): US 97916628 A 19970822
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6081774	A		41	G06F-017/27	

Abstract (Basic): US 6081774 A

NOVELTY - A non-real time development system (102) and a real time **retrieval system** (104) morphologically, syntactically and linguistically analyze a language based database and natural language query, respectively to generate one or more database keywords and query keywords, respectively. The database and query keywords represent content of language based database and natural language query (160), respectively.

DETAILED DESCRIPTION - The non-real time development system creates a database index (130) having one or more content based keywords of the database, automatically. The real time **retrieval system searches** the index for query keywords derived from natural language query based on user's queries. The non-real time development system comprises a software developer's kit for creating database index, utilizing a pattern dictionary that includes synonyms and skipwords. A morphous syntactic dictionary in the system includes morphological and syntactic information for words in the natural language of language based database and natural language query. The real time **retrieval system** has a natural language interface (170) that creates one or more query keywords utilizing pattern and morphosyntactic dictionaries. A query index matcher **matches one** or more query keywords with one or more database keywords.

USE - For retrieving information from language based database using natural language queries in Internet and intranet.

ADVANTAGE - Enables any software developer to add information **retrieval system** to pre-existing software application to provide a user interface that enables user to develop a query in natural language. The software developer's kit enables software developers to add natural language interface and associated information retrieval capability to existing software application without any development work.

DESCRIPTION OF DRAWING(S) - The figure shows functional block diagram of information **retrieval system** .

Non-real time development system (102)
Real time **retrieval system** (104)
Database index (130)
Natural language query (160)
Natural language interface (170)
pp; 41 DwgNo 1/19

Title Terms: INFORMATION; RETRIEVAL; SYSTEM; NATURAL; LANGUAGE; QUERY; LANGUAGE; BASED; DATABASE; NATURAL; LANGUAGE; QUERY; GENERATE; DATABASE; KEYWORD; QUERY; KEYWORD; RESPECTIVE
Derwent Class: T01
International Patent Class (Main): **G06F-017/27**
International Patent Class (Additional): **G06F-007/00**

File Segment: EPI

```
Set     Items   Description
S1    6282204   PARTIAL? OR FUZZY? OR PORTION? OR SIGNIFICANT? OR PORTION?
                OR FRACTION? OR FRAGMENT?
S2    5036189   MATCH? OR QUER? OR SEARCH? OR RETRIEV? OR LOCAT? OR IDENTI-
                F?
S3    9686031   STRING? OR SEARCHSTRING? OR CHARACTER? OR ALPHANUMERIC? OR
                LETTER? OR WORD? OR TERM? OR PHRASE?
S4      12113   (SINGLE OR ONE OR INDIVIDUAL? OR UNIQUE?)(N)(OCCUR? OR APP-
                EAR? OR MATCH?)
S5     189546   S2(N)(ENGINE? OR SOFTWARE? OR APPLICATION? OR SYSTEM? OR P-
                ROGRAM? OR CRAWLER? OR IA OR BOT OR ROBOT OR AGENT? OR TOOL?)-
                OR SEARCHENGINE?
S6        562   S1 AND S2 AND S3 AND S4
S7         30   S1 AND S5 AND S4
S8         53   S1(5N)S2 AND S6
S9         24   S3(2N)S1 AND S6
S10        24   S1(2N)S2 AND S6
S11        72   S7 OR S9 OR S10
S12        56   RD (unique items)
S13        44   S12 NOT PY>2001
S14        43   S13 NOT PD=20010405:20030405
S15        43   S14 NOT PD=20030405:20040409
S16        43   S15 NOT CY>2001
File   8:Ei Compendex(R) 1970-2004/Mar W4
         (c) 2004 Elsevier Eng.  Info. Inc.
File  35:Dissertation Abs Online 1861-2004/Mar
         (c) 2004 ProQuest Info&Learning
File 202:Info. Sci. & Tech. Abs. 1966-2004/Feb 27
         (c) 2004 EBSCO Publishing
File  65:Inside Conferences 1993-2004/Apr W1
         (c) 2004 BLDSC all rts. reserv.
File   2:INSPEC 1969-2004/Mar W4
         (c) 2004 Institution of Electrical Engineers
File  94:JICST-EPlus 1985-2004/Mar W3
         (c)2004 Japan Science and Tech Corp(JST)
File 111:TGG Natl.Newspaper Index(SM) 1979-2004/Apr 09
         (c) 2004 The Gale Group
File 233:Internet & Personal Comp. Abs. 1981-2003/Sep
         (c) 2003 EBSCO Pub.
File   6:NTIS 1964-2004/Apr W1
         (c) 2004 NTIS, Intl Cpyrght All Rights Res
File 144:Pascal 1973-2004/Mar W4
         (c) 2004 INIST/CNRS
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 1998 Inst for Sci Info
File  34:SciSearch(R) Cited Ref Sci 1990-2004/Apr W1
         (c) 2004 Inst for Sci Info
File  62:SPIN(R) 1975-2004/Feb W3
         (c) 2004 American Institute of Physics
File  99:Wilson Appl. Sci & Tech Abs 1983-2004/Mar
         (c) 2004 The HW Wilson Co.
```

16/5/6 (Item 6 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

00963649 E.I. Monthly No: EI8011083324 E.I. Yearly No: EI80044498
Title: PARTIAL - MATCH RETRIEVAL **IN AN INDEX SEQUENTIAL DIRECTORY.**
Author: Zvegintzov, N.
Source: Computer Journal v 23 n 1 Feb 1980 p 37-40
Publication Year: 1980
CODEN: CMPJA6 ISSN: 0010-4620
Language: ENGLISH
Journal Announcement: 8011
Abstract: An algorithm is described which, given an index sequential directory of keys, and given a set of **partially** specified templates, **retrieves** all keys in the directory that **match** **one** or more templates. Algorithms are given for the common special case where the keys are fixed length **strings** in lexicographic order. The origins, applications, and properties of these algorithms are discussed. 7 refs.
Descriptors: INFORMATION **RETRIEVAL** **SYSTEMS**
Classification Codes:
723 (Computer Software); 901 (Engineering Profession)
72 (COMPUTERS & DATA PROCESSING); 90 (GENERAL ENGINEERING)

16/5/7 (Item 1 from File: 35)
DIALOG(R)File 35:Dissertation Abs Online

01736162 ORDER NO: AADAA-I9963871
Use of genetic algorithms in information retrieval: Adapting matching functions

Author: Pathak, Praveen A.
Degree: Ph.D.
Year: 2000
Corporate Source/Institution: The University of Michigan (0127)
Chair: Michael Gordon
Source: VOLUME 61/03-A OF DISSERTATION ABSTRACTS INTERNATIONAL. PAGE 804. 141 PAGES
Descriptors: INFORMATION SCIENCE ; COMPUTER SCIENCE ; ARTIFICIAL INTELLIGENCE
Descriptor Codes: 0723; 0984; 0800

Information **retrieval systems** are complex in nature due to the interactions of document, query, and matching subsystems involved in the process of retrieval. Researchers have applied probabilistic, knowledge-based, and, more recently, artificial intelligence based techniques like neural networks and symbolic learning to this problem. Very few researchers have tried to use evolutionary algorithms like genetic algorithms (GA's). Previous attempts at using GA's have concentrated on modifying the document representations or modifying the query representations.

In this research, we explore the possibility of applying GA's to adapt the matching functions used in retrieval. We have described a method where an overall matching function is achieved by combining the results of the **individual matching** functions. The weights associated with **individual matching** functions have been adapted using GA's. We tested the method on two document collections. Experiments on these collections suggest that a GA based matching function adaptation **significantly** improves retrieval performance compared to the performance obtained by the best **individual matching** function.

We believe the promising outcomes of the GA based matching function adaptation merits continuing research. We briefly present possible areas of future research such as simultaneous adaptations of the three subsystems involved in retrieval, user profiling using this approach, and evolving new matching functions.

16/5/17 (Item 5 from file: 2)
DIALOG(R)File 2:INSPEC

4960073 INSPEC Abstract Number: C9507-1340F-017
Title: Algorithmic aspects of fuzzy **control**
Author(s): Koczy, L.T.
Author Affiliation: Dept. of Telecommun. & Telematics, Tech. Univ. Budapest, Hungary
Journal: International Journal of Approximate Reasoning vol.12, no.3-4 p.159-219
Publication Date: April-May 1995 Country of Publication: USA
CODEN: IJARE4 ISSN: 0888-613X
U.S. Copyright Clearance Center Code: 0888-613X/95/$9.50
Language: English Document Type: Journal Paper (JP)
Treatment: Theoretical (T)
Abstract: **Fuzzy** control is still the most important application of **fuzzy** theory. It is a generalized form of expert control using **fuzzy** sets in the definition of vague/linguistic predicates, modeling a system by If...then rules. In classical approaches the essential idea is that a fact (observation) which is known concerning the actual state of the **system matches one** or several rules in the model to some positive degree, the conclusion is calculated by the evaluation of the degree of these **matches** , and the **matched** rules themselves. In these approaches, the rules contain linguistic **terms** , i.e., **fuzzy** sets in the consequent parts, and these **terms** , weighted with their respective degrees of **matching** , are combined in order to obtain a **fuzzy** conclusion from which the crisp action is obtained by defuzzification as e.g. the center of gravity method. The paper summarizes these classical methods and turns attention to their weak point: the computational complexity aspect. As a **partial** solution, the use of sparse rule bases is proposed and rule interpolation as a fitting inference engine is presented. The problem of preserving or not preserving linearity is discussed when **terms** in the rules are restricted to piecewise linear. (53 Refs)
Subfile: C
Descriptors: computational complexity; computational linguistics; **fuzzy** control; **fuzzy** set theory; inference mechanisms; interpolation; piecewise-linear techniques
Identifiers: **fuzzy** control; **fuzzy** theory; algorithmic aspects; expert control; **fuzzy** sets; vague predicates; linguistic predicates; if...then rules; actual system state; **matched** rules; linguistic **terms** ; **fuzzy** conclusion; crisp action; defuzzification; computational complexity; **partial** solution; sparse rule bases; rule interpolation; inference engine; linearity; piecewise linear rules
Class Codes: C1340F (Fuzzy control); C4240C (Computational complexity); C4130 (Interpolation and function approximation); C6170K (Knowledge engineering techniques); C3230 (Control logic); C4210L (Formal languages and computational linguistics); C1310 (Control system analysis and synthesis methods); C1160 (Combinatorial mathematics)

16/5/21 (Item 9 from file: 2)
DIALOG(R)File 2:INSPEC

01847591 INSPEC Abstract Number: C82019824
Title: Freud or fraud. A computer psychiatrist that is as insulating as it is amusing
Author(s): Chappel, B.
Journal: Microcomputer Printout vol.2, no.9 p.40-1, 51, 71
Publication Date: Oct. 1981 Country of Publication: UK
CODEN: MPRIDB ISSN: 0261-4499
Language: English Document Type: Journal Paper (JP)
Treatment: Applications (A)
Abstract: The program attempts to simulate the art of a psychoanalyst by conducting an interview with the player ('patient'). The statement is broken down into separate **words** which are then compared one by one against a key list of **words** and **phrases** . An alternative method would do a sliding **string** **search** along the statement, on each pass of the statement **matching** **one** key **word** or **phrase** against each **string** **portion** of the statement. The method used in this program gives a response time of from 1 to 4 seconds. (0 Refs)
Subfile: C
Descriptors: medical diagnostic computing; personal computing
Identifiers: computer psychiatrist; interview; list of **words** and **phrases** ; sliding **string** **search** ; response time
Class Codes: C7330 (Biology and medicine); C7830 (Home computing)

```
Set     Items   Description
S1     6282204   PARTIAL? OR FUZZY? OR PORTION? OR SIGNIFICANT? OR PORTION?
                 OR FRACTION? OR FRAGMENT?
S2     5036189   MATCH? OR QUER? OR SEARCH? OR RETRIEV? OR LOCAT? OR IDENTI-
                 F?
S3     9686031   STRING? OR SEARCHSTRING? OR CHARACTER? OR ALPHANUMERIC? OR
                 LETTER? OR WORD? OR TERM? OR PHRASE?
S4       12113   (SINGLE OR ONE OR INDIVIDUAL? OR UNIQUE?)(N)(OCCUR? OR APP-
                 EAR? OR MATCH?)
S5      189546   S2(N)(ENGINE? OR SOFTWARE? OR APPLICATION? OR SYSTEM? OR P-
                 ROGRAM? OR CRAWLER? OR IA OR BOT OR ROBOT OR AGENT? OR TOOL?)-
                 OR SEARCHENGINE?
S6         562   S1 AND S2 AND S3 AND S4
S7          30   S1 AND S5 AND S4
S8          53   S1(5N)S2 AND S6
S9          24   S3(2N)S1 AND S6
S10         24   S1(2N)S2 AND S6
S11         72   S7 OR S9 OR S10
S12         56   RD (unique items)
S13         44   S12 NOT PY>2001
S14         43   S13 NOT PD=20010405:20030405
S15         43   S14 NOT PD=20030405:20040409
S16         43   S15 NOT CY>2001
S17    4211611   WORD? OR PHRASE? OR TERM? OR CHARACTER()STRING?
S18      17297   S1(N)S17
S19         12   S18 AND S4
S20          5   S19 NOT S11
S21          4   RD (unique items)
S22          4   S21 NOT PY>2001
File   8:Ei Compendex(R) 1970-2004/Mar W4
         (c) 2004 Elsevier Eng.  Info. Inc.
File  35:Dissertation Abs Online 1861-2004/Mar
         (c) 2004 ProQuest Info&Learning
File 202:Info. Sci. & Tech. Abs. 1966-2004/Feb 27
         (c) 2004 EBSCO Publishing
File  65:Inside Conferences 1993-2004/Apr W1
         (c) 2004 BLDSC all rts. reserv.
File   2:INSPEC 1969-2004/Mar W4
         (c) 2004 Institution of Electrical Engineers
File  94:JICST-EPlus 1985-2004/Mar W3
         (c)2004 Japan Science and Tech Corp(JST)
File 111:TGG Natl.Newspaper Index(SM) 1979-2004/Apr 09
         (c) 2004 The Gale Group
File 233:Internet & Personal Comp. Abs. 1981-2003/Sep
         (c) 2003 EBSCO Pub.
File   6:NTIS 1964-2004/Apr W1
         (c) 2004 NTIS, Intl Cpyrght All Rights Res
File 144:Pascal 1973-2004/Mar W4
         (c) 2004 INIST/CNRS
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 1998 Inst for Sci Info
File  34:SciSearch(R) Cited Ref Sci 1990-2004/Apr W1
         (c) 2004 Inst for Sci Info
File  62:SPIN(R) 1975-2004/Feb W3
         (c) 2004 American Institute of Physics
File  99:Wilson Appl. Sci & Tech Abs 1983-2004/Mar
         (c) 2004 The HW Wilson Co.
```

Search Results

Search Results for: **[search and and (minimal portion)]**
Found **5** of **130,565 searched.**

Search within Results

> Advanced Search
> Search Help/Tips

Sort by: Title Publication Publication Date **Score** Binder

Results 1 - 5 of 5 short listing

1 Energy management for battery-powered embedded systems 77%
Daler Rakhmatov , Sarma Vrudhula
ACM Transactions on Embedded Computing Systems (TECS) August 2003
Volume 2 Issue 3

Portable embedded computing systems require energy autonomy. This is achieved by batteries serving as a dedicated energy source. The requirement of portability places severe restrictions on size and weight, which in turn limits the amount of energy that is continuously available to maintain system operability. For these reasons, efficient energy utilization has become one of the key challenges to the designer of battery-powered embedded computing systems.In this paper, we first present a novel a ...

2 Middleware for context sensitive mobile applications 77%
K. A. Hawick , H. A. James
Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21 January 2003

Contextual information such as spatial location can significantly enhance the utility of mobile applications. We introduce the concept of active preferences that represent a combination of user preference information and choices combined with spatial or temporal information. Active preferences set the policy on how a mobile application should customise its behaviour not just for a particular user but as that user moves to different locations and interacts with other mobile users or with fixed lo ...

3 DBMS implementation experience: FLASH: a language-independent, portable file access system 77%
James E. Allchin , Arthur M. Keller , Gio Wiederhold
Proceedings of the 1980 ACM SIGMOD international conference on Management of data May 1980

A file access system, flash, for use in building database systems is described. It supports access from several languages, including pascal, fortran, and interlisp. Flash provides record level access to a file with multiple indexes using symbolic keys. It is portable and written in Pascal with support routines in dec System 20 macro. The file

access system is designed to run on computers of various sizes and capabilities, including micros. Co ...

4 Piranha: a scalable architecture based on single-chip multiprocessing 77%
Luiz André Barroso , Kourosh Gharachorloo , Robert McNamara , Andreas Nowatzyk , Shaz Qadeer , Barton Sano , Scott Smith , Robert Stets , Ben Verghese
ACM SIGARCH Computer Architecture News , Proceedings of the 27th annual international symposium on Computer architecture May 2000
Volume 28 Issue 2

The microprocessor industry is currently struggling with higher development costs and longer design times that arise from exceedingly complex processors that are pushing the limits of instruction-level parallelism. Meanwhile, such designs are especially ill suited for important commercial applications, such as on-line transaction processing (OLTP), which suffer from large memory stall times and exhibit little instruction-level parallelism. Given that commercial applications constitute by fa ...

5 Principles for writing reusable libraries 77%
Glenn S. Fowler , David G. Korn , Kiem-Phong Vo
ACM SIGSOFT Software Engineering Notes , Proceedings of the 1995 Symposium on Software reusability August 1995
Volume 20 Issue SI

Over the past 10 years, the Software Engineering Research Department in AT&T has been engaging in a research program to build a collection of highly portable advanced software tools known as Ast, Advanced Software Technology. A recent monograph, "Practical Reusable UNIX Software" (John Wiley & Sons, Inc., 1995), summarizes the philosophy and components of this research program. A major component of this program is a collection of portable, and reusable libraries servicin ...

Results 1 - 5 of 5 short listing

Citation

International Conference on Management of Data >archive

Proceedings of the 1980 ACM SIGMOD international conference on Management of data >toc

1980 , Santa Monica, California

SESSION: Query processing >toc

Some properties of Cartesian product files

Authors

C. C. Chang National Chung Hsin University, Taichung, Taiwan
R. C. T. Lee National Tsing Hua University, Hsinchu, Taiwan
H. C. Du University of Washington, Seattle, Washington

Sponsor

: undetermined

Publisher

ACM Press New York, NY, USA

Pages: 157 - 168 Series-Proceeding-Section-Article
Year of Publication: 1980
ISBN:0-89791-018-4

doi http://doi.acm.org/10.1145/582250.582276 (Use this link to Bookmark this page)

> full text > abstract > references > citings

> Discuss > Similar > Review this Article Save to Binder

> BibTex Format

FULL TEXT: Access Rules

pdf **1.03 MB**

ABSTRACT

In this paper, we first introduced the concept of Cartesian product files. We then derived a formula for random files. A computer simulation experiment was performed to compare these two files. So far as shown by the experimental results, the Cartesian product file concept was indeed a good one. We also showed that the problem of designing an optimal Cartesian product file was partially

related to the problem of finding a minimal N-tuple. A method to find minimal N-tuples was presented and its properties were discussed.

REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 Bently, J. L. (1979): Nultidimensional Binary Search Trees in Database Applications, IEEE Trans. on Software Engineering, Vol. SE-5, No. 4, July 1979, pp. 333--342.

2 Lee, R. C. T. and Tseng, S. H. (1979): Multi-key Sorting, Policy Analysis and Information Systems, Vol. 3, No. 2, Dec. 1979, pp. 1--20.

3 Lin, W. C., Lee, R. C. T. and Du. H. C. (1979); Common Properties of Some Multi-attribute File Systems, IEEE Trans. on Software Engineering, Vol. SE-5, No. 2, March 1979, pp. 160--174.

4 Liou, J. H. and Yao, S. B. (1977): Multi-dimensional Clustering for Data Base Organizations, Information Systems, Vol. 2, 1977, pp. 187--198.

5 Rivest, R. L. (1976): Partial-match Retrieval Algorithms, SIAM J. Comput., Vol. 15, No. 1, March 1976. pp. 19--50.

6 James B. Rothnie, Jr. , Tomas Lozano, Attribute based file organization in a paged memory environment, Communications of the ACM, v.17 n.2, p.63-69, Feb. 1974

7 Slagle, J. R., Chang, C. L. and Lee, R. C. T. (1974): Experiments with Some Clustering Analysis Algorithm, Pattern Recognition, Vol. 6, 1974.

CITINGS

Edward Omiecinski , Peter Scheuermann, A global approach to record clustering and file reorganization, Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval, p.201-219, July 02-06, 1984, Cambridge, England

Some Properties of Cartesian Product Files

C. C. Chang, R. C. T. Lee and H. C. Du

C. C. Chang is with the Dept. of Applied Mathematics, National Chung Hsin University, Taichung, Taiwan. R. C. T. Lee is with the Inst. of Computer and Decision Sciences, National Tsing Hua University, Hsinchu, Taiwan. H. C. Du is with the Dept. of Computer Science, University of Washington, Seattle, Washington, U. S. A.

This research was supported in part by the National Science Council, Republic of China, under contract NSC-68-0404-03(06).

Abstract

In this paper, we first introduced the concept of Cartesian product files. We then derived a formula for random files. A computer simulation experiment was performed to compare these two files. So far as shown by the experimental results, the Cartesian product file concept was indeed a good one. We also showed that the problem of designing an optimal Cartesian product file was partially related to the problem of finding a minimal N-tuple. A method to find minimal N-tuples was presented and its properties were discussed.

Section 1. Introduction

In this paper, we are concerned with the problem of designing optimal multi-attribute file systems for partial match queries [Rivest 1976, Rothnie and Lozano 1974, Liou and Yao 1975, Bentley 1979, Lee and Tseng 1979, Lin, Lee and Du 1979]. By a multi-attribute file system, we mean a file system whose records are characterized by more than one attribute. By partial match queries, we mean queries of the following form: Retrieve all records where $A_{i_1}=a_{i_1}$, $A_{i_2}=a_{i_2}$, ..., $A_{i_j}=a_{i_j}$ and $i_1 \neq i_2 \neq \ldots \neq i_j$.

We shall assume that every file is divided into buckets. The problem of multi-attribute file design can be explained by considering the two file systems shown in Table 1.1 and Table 1.2.

Table 1.1 here

Table 1.2 here

In both tables, a query (a,*) denotes a query retrieving records with the first attribute equal to a and the second attribute with any value. Similarly, for a 3-attribute file system, a query denoted as (*,b,c) denotes a query retrieving all records with A_2=b and A_3=c and A_1 can be of any value. The reader can see that the average number of buckets to be examined, over all possible queries, is 2 for the file system in Table 1.2 and 4 for that in Table 1.1.

Thus the problem of multi-attribute file system design for partial match queries is as follows: Given a set of multi-attribute records, arrange the records into the NB buckets in such a way that the average number of buckets to be examined, over all possible partial match queries, is minimized.

The general problem stated above is rather hard to solve. In this paper, we shall limit ourselves to the case where all possible records are present. Note that every record is characterized by N attributes A_1, A_2, A_3, ..., A_N. Let the domain of attribute A_i be denoted as D_i. Thus the set of all possible records is $D_1 \times D_2 \times \ldots \times D_N$. In the rest of this paper, whenever we discuss the partial match problem, we shall assume that every possible record in this set $D_1 \times D_2 \times \ldots \times D_N$ is present. If some of the records in the set $D_1 \times D_2 \times \ldots \times D_N$ are missing, we consider the optimization of Cartesian product files with respect to partial match patterns which were defined by Lin, Lee and Du [1979].

Section 2. Cartesian Product Files and Random Files

Multi-attribute file system design for partial match queries has been considered by many authors. Rivest [1976] suggested the string homomorphism hashing (SHH for short) method. Rothnie and

© 1980 ACM 0-89791-018-4/80/0500/0157 $00.75

Lozano [1974] suggested the multi-key hashing (MKH for short) method. Liou and Yao [1975] suggested the multi-dimensional directory (MDD for short) method. Lee and Tseng [1979] suggested the multi-key sorting (MKS for short) method.

In [Lin, Lee and Du 1979], it was proved that all of those file designing methods exhibit one common property: Records in one bucket are similar to one another. In [Lin, Lee and Du 1979], it was also pointed out that the file system designed by using the SHH, MKH and MDD methods are all Cartesian product files which are defined as follows.

Definition: Let there be N attributes A_1, A_2, ..., A_N. Let the domain of A_i be D_i. A Cartesian product file is a file in which the records in every bucket is of the form $D_{1s} \times D_{2s} \times \ldots \times D_{Ns}$ where D_{is} is a subset of D_i.

Example 2.1

Let $D_1=\{a,b,c,d\}=D_2$. Let $D_{11}=\{a,b\}$ $=D_{21}$. Let $D_{12}=\{c,d\}=D_{22}$. Then the following file is a Cartesian product file.

Bucket 1: $D_{11}\times D_{21}=\{(a,a), (a,b), (b,a), (b,b)\}$
Bucket 2: $D_{11}\times D_{22}=\{(a,c), (a,d), (b,c), (b,d)\}$
Bucket 3: $D_{22}\times D_{21}=\{(c,a), (c,b), (d,a), (d,b)\}$
Bucket 4: $D_{12}\times D_{22}=\{(c,c), (c,d), (d,c), (d,d)\}$

The reader can see that the above file system is exactly the same file system shown in Table 1.2.

Example 2.2

Let $D_1=\{a,b,c,d,e\}$ and $D_2=\{a,b,c,d\}$. Let $D_{11}=\{a,b,c\}$, $D_{12}=\{d,e\}$, $D_{21}=\{a,b\}$ and $D_{22}=\{c,d\}$. Then the following file system is a Cartesian product file system.

Bucket 1: $D_{11}\times D_{21}=\{(a,a), (a,b), (b,a), (b,b), (c,a), (c,b)\}$
Bucket 2: $D_{11}\times D_{22}=\{(a,c), (a,d), (b,c), (b,d), (c,c), (c,d)\}$
Bucket 3: $D_{12}\times D_{21}=\{(d,a), (d,b), (e,a), (e,b)\}$
Bucket 4: $D_{12}\times D_{22}=\{(d,c), (d,d), (e,c), (e,d)\}$

Note that in this case, the number of records in Bucket 1 is not the same as that in Bucket 3.

It was also pointed out in [Lin, Lee and Du 1979] that records in a Cartesian product file form a short spanning path [Slagle, Chang and Lee 1974]. That is, records in a bucket of a Cartesian product file can be ordered into a sequence R_1, R_2, ..., R_{BZ} and for every pair of consecutive records R_i and R_{i+1} ($1 \le i < BZ$), these two records are different at only one attribute. For instance, consider Bucket 1 in the above example. The records in this bucket can be reduced into the following sequence:

(a,a)
(a,b)
(b,b)
(b,a)
(c,a)
(c,b)

Since two consecutive records are different at only one attribute, a Cartesian product file exhibits the property of clustering similar records together.

It is our conjecture that the Cartesian product file concept is optimal in the sense that a Cartesian product file is always better than a non-Cartesian product file. We have not been able to prove this conjecture yet. However, we do have some results to show the superiority of Cartesian product files.

Let us call a file where records are randomly placed in buckets a random file. In the following, we shall derive a formula giving the expected number of buckets accessed over all possible partial match queries for a random file. Again, let us assume that our records are characterized by N attributes A_1, A_2, ..., A_N and the domain of A_i is D_i. Let the number of elements in D_i be denoted as d_i. Then the number of records NR is equal to $d_1 d_2 \ldots d_N$. Let NB denote the number of buckets. Then the bucket size BZ is equal to NR/NB. Let ANB_R denote the expected number of buckets being accessed over all possible partial match queries in a random file.

First let us consider a special partial match query $A_i=a_i$ where $a_i \in D_i$. There are $d_1\times d_2\times\ldots\times d_{i-1}\times d_{i+1}\times\ldots\times d_N$ records satisfying the condition $A_i=a_i$. Since each record is randomly assigned to a bucket, the probability that a bucket receives a record is 1/NB. The expected number of buckets being accessed for this query is equal to the number of buckets which are not empty when we randomly assign $d_1\times d_2\times\ldots\times d_{i-1}\times d_{i+1}\times\ldots\times d_N$ records to NB buckets.

The probability of a bucket being empty

$$= \left(1-\frac{1}{NB}\right)^{d_1\cdot d_2\cdots d_{i-1}\cdot d_{i+1}\cdots d_N}$$

= the probability that all $d_1\cdot d_2\cdots d_{i-1}\cdot d_{i+1}\cdot d_N$ records are assigned to other buckets.

The probability that a bucket is not empty

$$= 1-\left(1-\frac{1}{NB}\right)^{d_1\cdot d_2\cdots d_{i-1}\cdot d_{i+1}\cdots d_N}$$

The expected number of buckets which are not empty

$$= NB(1-(1-\frac{1}{NB})^{d_1 \cdot d_2 \cdots d_{i-1} \cdot d_{i+1} \cdots d_N}).$$

Note that for all $a_i \varepsilon D_i$, all partial match queries $A_i = a_i$ produce the same result.

ANB_R = (Σ the expected number of all partial match queries buckets being accessed for a partial match query)/the total number of different partial match queries

The total number of partial match queries can be found as follows:

(1) There are $d_1+d_2+\ldots+d_N$ partial match queries which involve exactly one attribute.

(2) There are $d_1d_2+d_1d_3+\ldots+d_{N-1}d_N$ partial match queries which involve exactly two attributes.

(3) There are $d_1d_2\ldots d_{N-1}+\ldots+d_2d_3\ldots d_N$ partial match queries which involve exactly N-1 attributes.

Let $\{d_{I_1}, d_{I_2}, \ldots, d_{I_i}\}$ be a subset with i elements chosen from $\{d_1, d_2, \ldots, d_N\}$. In general, there are

$$\sum_{\{d_{I_1}, d_{I_2} \ldots, d_{I_i}\} \varepsilon \{d_1, d_2, \ldots, d_N\}} d_{I_1} \cdot d_{I_2} \cdots d_{I_i}$$

partial match queries which involve exactly i attributes. The total number of queries

$$= d_1+d_2+\ldots+d_N$$
$$+ d_1d_2+d_2d_3+\ldots+d_{N-1}d_N$$
$$+ \ldots$$
$$+ d_1d_2\cdots d_{N-1}+\ldots+d_2d_3\cdots d_N$$
$$= \sum_{i=1}^{N-1} \sum_{\{d_{I_1}, d_{I_2}, \ldots, d_{I_i}\} \varepsilon \{d_1, d_2, \ldots, d_N\}} d_{I_1} \cdot d_{I_2} \cdot d_{I_3} \cdots d_{I_i}$$

Let TNB_i be the total number of buckets being accessed over all the partial match queries with i attributes being specified.

(1) TNB_1

$$= \sum_{d_i \varepsilon \{d_1, d_2, \ldots, d_N\}} d_i \cdot NB(1-(1-\frac{1}{NB})^{d_1 \cdot d_2 \cdots d_{i-1} \cdot d_{i+1} \cdots d_N})$$

(2) TNB_2

$$= \sum_{\{d_i, d_j\} \varepsilon \{d_1, d_2, \ldots, d_N\} \text{ and } i<j} d_i \cdot d_j \cdot NB(1-(1-\frac{1}{NB})^{(d_1 \cdot d_2 \cdots d_{N-1} \cdot d_N)/(d_i \cdot d_j)})$$

(3) TNB_{N-1}

$$= d_2 \cdot d_3 \cdots d_N \cdot NB(1-(1-\frac{1}{NB})^{d_1})$$
$$+ d_1 \cdot d_3 \cdots d_N \cdot NB(1-(1-\frac{1}{NB})^{d_2})$$
$$+ d_1 \cdot d_2 \cdots d_{N-1} \cdot NB(1-(1-\frac{1}{NB})^{d_N})$$

In general,

TNB_i

$$= \sum_{\{d_{I_1}, d_{I_2}, \ldots, d_{I_i}\} \varepsilon \{d_1 \cdot d_2 \cdots d_N\} \; I_1<I_2<I_i} d_{I_1} \cdot d_{I_2} \cdots d_{I_i} \cdot NB(1-(1-\frac{1}{NB})^{(d_1 \cdot d_2 \cdots d_N)/(d_{I_1} \cdot d_{I_2} \cdots d_{I_i})})$$

and

$$ANB_R = \sum_{i=1}^{N} TNB_i/(d_1+d_2+\ldots+d_N+d_1d_2+\ldots+d_{N-1}d_N+\ldots+d_2d_3\ldots d_N)$$

Hence given $d_1, d_2, \ldots, d_N$, NB and $NR = d_1 \cdot d_2 \cdots d_N$, $BZ = NR/NB$, we can calculate ANB_R.

<u>Example 2.1</u>

Let d_1, d_2, d_3 be 3, 4 and 5 respectively. Let NB be equal to 4. In this case,

$$TNB = 3\times4\times(1-(1-\tfrac{1}{4})^{4\times5})$$
$$+ 4\times4\times(1-(1-\tfrac{1}{4})^{3\times5})$$
$$+ 5\times4\times(1-(1-\tfrac{1}{4})^{3\times4})$$
$$+ 3\times4\times4\times(1-(1-\tfrac{1}{4})^{5})$$
$$+ 3\times5\times4\times(1-(1-\tfrac{1}{4})^{4})$$
$$+ 4\times5\times4\times(1-(1-\tfrac{1}{4})^{3})$$
$$= 170.9868.$$

$$ANB_R = 170.9868/(3+4+5+3\times4+4\times5+3\times5)$$
$$= 170.9868/59$$
$$= 2.8981.$$

We have derived the formula for the expected number of buckets to be accessed over all possible partial match queries. In the next sections, we shall derive similar formulas for Caretesian product files. We hope that through these formulas, we can show the superiority of random files. We are still working on this proof. That we still can not prove it is probably due to the fact that the formula for random files is extremely messy.

To test our conjecture, we conducted a computer simulation experiment.

The purpose of this experiment was to compare the performances of Cartesian product files and random files. We used the Monte Carlo simulation method. Thirty sets of data were generated. Each set of data was characterized by two, three or four keys. A random number generator was first used to generate the number d_i which was the number of elements in the domain of the ith key. Then the number NR (the number of records) was calculated according to the following formula:

$$NR = d_1d_2d_3d_4.$$

The same random number generator was used

to generate NB, the number of buckets, under the constraint that NR/NB was an integer.

We then calculated the bucket size BZ according to the formula

$$BZ = NR/NB.$$

For each data set, we calculated the average number of buckets accessed, over all possible partial match queries, if the Cartesian product file concept was used. This number is denoted as ANB_{CP}. The method of obtaining this number will be explained in later sections. For each set, the corresponding ANB_R was also calculated using the formula derived in this section. The result is shown in Table 2.1. From the experimental results

---

Table 2.1 here

---

obtained thus far, it can be seen that Cartesian product files are indeed better than random files.

Section 3. The Designing of Optimal Cartesian Product Files

If a file is a Cartesian product file, for every bucket, records in this bucket are of the form of

$$D_{1s} \times D_{2s} \times \ldots \times D_{Ns}.$$

Let the domain size of D_{is} be denoted as z_i. To simplify our discussion, we shall assume that z_i is the same for every bucket. Note that this is not the case for the file shown in Example 2.2. In that case, $z_1=3$ for Bucket 1 and $z_1=2$ for Bucket 3. It is much more complicated to design such an optimal file.

For a Cartesian product file, to minimize the average number of buckets to be examined over all possible partial match queries, we may simply try to minimize the total number of queries which need to examine a bucket in the file. (Note that this number is the same for all buckets in a Cartesian product file.) We now ask, what is the number of partial match queries which need to examine a bucket in a Cartesian product file? The answer is as follows.

(1) There are $z_1+z_2+\ldots+z_N$ partial match queries which involve exactly one attribute.
(2) There are $z_1z_2+z_1z_3+\ldots+z_{N-1}z_N$ partial match queries which involve exactly two attributes.
(3) There are $z_1z_2\ldots z_{N-1}+\ldots+z_2z_3\ldots z_N$ partial match queries which involve exactly N-1 attributes.

Totally, for each bucket in a Cartesian product file, the total number of partial match queries which need to examine this bucket is

$$\begin{aligned} & z_1+\ldots+z_N \\ + & z_1z_2+\ldots+z_{N-1}z_N \\ + & \ldots \\ + & z_1z_2\ldots z_{N-1}+\ldots+z_2z_3\ldots z_N \end{aligned}$$

Let us now state formally the problem of designing an optimal Cartesian product file as follows.

We assume that each record is characterized by N attributes $A_1,A_2,\ldots,A_N$ and the domain of A_i is D_i. The size of D_i is d_i. There are totally $d_1d_2\ldots d_N$ records present. The number of buckets is NB. The bucket size is therefore $(d_1d_2\ldots d_N)/NB=C$ (C is an integer).

Our problem is to find $z_1, z_2, \ldots, z_N$ satisfying the following conditions:

(1) $z_1, z_2 \ldots$ and z_N are integers.
(2) $z_1z_2 \ldots z_N = C$.
(3) $d_i/z_i=m_i$=an integer (This means that each domain D_i is divided into m_i equal subsets, where the size of each subset is z_i.)
(4) $$\begin{aligned} & z_1+z_2+\ldots+z_N \\ + & z_1z_2+\ldots+z_{N-1}z_N \\ + & \ldots \\ + & z_1z_2\ldots z_{N-1}+\ldots+z_2z_3\ldots z_N \text{ is} \end{aligned}$$
minimized over all possible $(z_1, z_2, \ldots, z_N)$'s satisfying (1), (2) and (3).

<u>Example 3.1</u>

Consider the case where

$d_1 = 8$
$d_2 = 4$
$d_3 = 9$
and $NB = 6$

In this case, the bucket size is $(8\times4\times9)/6=48$. There are two feasible solutions satisfying the first three conditions. The first one is: $z_1=8$, $z_2=2$, and $z_3=3$. The second one is $z_1=4$, $z_2=4$, and $z_3=3$.

For the first solution,

$$\begin{aligned} & z_1+z_2+z_3+z_1z_2+z_1z_3+z_2z_3 \\ = & 8+2+3+8\times2+8\times3+2\times3 \\ = & 59. \end{aligned}$$

For the second solution,

$$\begin{aligned} & z_1+z_2+z_3+z_1z_2+z_1z_2+z_2z_3 \\ = & 4+4+3+4\times4+4\times3+4\times3 \\ = & 51. \end{aligned}$$

We therefore conclude that the second solution is the optimum solution. In this case,

$m_1=8/4=2$
$m_2=4/4=1$
$m_3=9/3=3$

Our Cartesian product file system

divides D_1 into two subsets: D_{11} and D_{12}, D_2 into one subset, and D_3 into three subsets: D_{31}, D_{32} and D_{33}. The six buckets are arranged as follows:

Bucket 1: $D_{11} \times D_2 \times D_{31}$
Bucket 2: $D_{11} \times D_2 \times D_{32}$
Bucket 3: $D_{11} \times D_2 \times D_{33}$
Bucket 4: $D_{12} \times D_2 \times D_{31}$
Bucket 5: $D_{12} \times D_2 \times D_{32}$
Bucket 6: $D_{12} \times D_2 \times D_{33}$

The reader may wonder how this optimum solution can be found. Since an integer can be factored into a finite number of different N-tuples, there are a finite number of feasible solutions, and we can conduct an exhaustive search. That is, given $z_1, \ldots, z_N$, we can calculate

$$\begin{aligned} & z_1+z_2+\ldots+z_N \\ + \; & z_1z_2+\ldots+z_{N-1}z_N \\ + \; & \ldots \\ + \; & z_1z_2\ldots z_{N-1}+\ldots+z_2z_3\ldots z_N \end{aligned}$$

We then choose the z_i's such that the above is minimized. However, we shall show that an exhaustive searching through all possible solutions can be avoided. Let us consider Example 3.1 again. The first solution of the problem is (8,2,3). In this 3-tuple, there exists a pair (8,2) which can be transformed into (4,4) ($8 \times 2 = 4 \times 4$) without affecting the feasibility of the solution. However, this transformation decreases not only the value of $z_1+z_2+z_3$ but also the value of $z_1z_2+z_1z_3+z_2z_3$.

For the second solution (4,4,3), there simply does not exist a pair (z_i,z_j) such that (z_i,z_j) can be transformed into (z_i',z_j') where $z_i'z_j'=z_iz_j$ and $z_i'+z_j'<z_i+z_j$.

Let us now consider the following problem: Given an N-tuple $(z_1,z_2,\ldots,z_N)$ where z_i's are all integers and $\prod_{i=1}^{N} z_i=C$, can we transform it into another N-tuple $(z_1',z_2',\ldots,z_N')$ such that z_i''s are all integers, $\prod_{i=1}^{N} z_i'=C$, but the value of

$$\begin{aligned} & z_1'+z_2'+\ldots+z_N' \\ + \; & z_1'z_2'+\ldots+z_{N-1}'z_N' \\ + \; & \ldots \\ + \; & z_1'z_2'\ldots z_{N-1}'+\ldots+z_2'z_3'\ldots z_N' \end{aligned}$$

is smaller than the value of

$$\begin{aligned} & z_1+z_2+\ldots+z_N \\ + \; & z_1z_2+\ldots+z_{N-1}z_N \\ + \; & \ldots \\ + \; & z_1z_2\ldots z_{N-1}+\ldots+z_2z_3\ldots z_N \end{aligned}$$

In the following section, we shall discuss this problem and its solution in detail.

Section 4. Some Theories of Minimal N-tuple

In the rest of this paper, whenever we mention an N-tuple $(a_1,a_2,\ldots,a_N)$, we shall assume that a_i is an integer. Without losing generality, whenever possible, we shall also assume that $a_i \le a_{i+1}$.

Definition:

An N-tuple $(a_1,a_2,\ldots,a_N)$ is called an N-tuple of C if $\prod_{i=1}^{N} a_i=C$.

Definition:

A 2-tuple (a_1,a_2) is called a minimal 2-tuple if for every other 2-tuple (a_1',a_2') where $a_1a_2=a_1'a_2'$, $a_1+a_2<a_1'+a_2'$.

Definition:

An N-tuple $(a_1,a_2,\ldots,a_N)$ is called a minimal N-tuple of C, if $\prod_{i=1}^{N} a_i=C$ and for $1 \le i, j \le N$, (a_i,a_j) is a minimal 2-tuple.

Example 4.1

(2,4,9) is not a minimal 3-tuple because (2,9) and (4,9) are not minimal 2-tuples. The 3-tuple (3,4,6) is a minimal 3-tuple because each pair in this 3-tuple is a minimal 2-tuple.

Definition:

Given an N-tuple $S=(a_1,a_2,\ldots,a_N)$, $F(S,K)$, $1 \le K \le N$, is defined as follows:

$$F(S,K) = \sum_{i_1<i_2<\ldots<i_K} a_{i_1}a_{i_2}\ldots a_{i_K}$$

for all possible $(i_1,i_2,\ldots,i_K)$'s

For instance,

$$\begin{aligned} F(S,1) &= a_1+a_2+\ldots+a_N \\ F(S,2) &= a_1a_2+a_1a_3+\ldots+a_{N-1}a_N. \\ &\;\vdots \\ F(S,N-1) &= a_1a_2\ldots a_{N-1}+\ldots+a_2a_3\ldots a_N. \end{aligned}$$

In the following, we shall present an algorithm which transforms an arbitrary N-tuple of C into a minimal N-tuple of C.

Algorithm A: An algorithm which transforms an N-tuple of C into a minimal N-tuple of C.

Input: $(a_1,a_2,\ldots,a_N)$ and $\prod_{i=1}^{N} a_i=C$.
Output: A minimal N-tuple of C.
Step 1: $I \leftarrow N$, $J \leftarrow N-1$.

Step 2: $A=a_J \cdot a_I$. Find (p,q) where (p,q) is a minimal 2-tuple of A.
Step 3: If $(p,q)=(a_J,a_I)$, go to Step 6.
Step 4: Reorder $(a_1,a_2,\ldots,p,q,\ldots,a_N)$. We obtain a new N-tuple $(a_1,a_2,\ldots,a_N)$, such that $a_{I-1} \le a_I$ for $I=2,\ldots,N$.
Step 5: Return to Step 1.
Step 6: If J is not equal to 1, $J \leftarrow J-1$ and return to Step 2.
Step 7: If I is not equal to 2, $I \leftarrow I-1$, $J \leftarrow I-1$ and return to Step 2.
Step 8: $(a_1,a_2,\ldots,a_N)$ is a minimal N-tuple of C.

Example 4.2

Input (34,35,105)

1) $I=3, J=2$.
2) $A=a_2 a_3=35\times105=3475$. We find (49,75) as the minimal 2-tuple of 3475.
3) Reorder. We obtain (34,49,75).
4) Return to Step 1.
5) $I=3, J=2$.
6) $A=a_2 \cdot a_3=49\times75=3475$. We find (49,75) is the minimal 2-tuple of 3475.
7) Go to Step 6. Since $J=2\neq1$, $J\leftarrow1$. Return to Step 2.
8) $A=a_1 \cdot a_3=34\times75=2550$. We find (50,51) as the minimal 2-tuple of 2550.
9) Reorder. We obtain (49,50,51). Since every 2-tuple in (49,50,51) is a minimal 2-tuple, (49,50,51) is the output.

Let us check into Example 4.2 again. The 3-tuples transformed are:

$$(34,35,105)\rightarrow(34,49,75)\rightarrow(49,50,51)$$

The F(S,1)'s corresponding to the above 3-tuple are 174, 158 and 150 respectively. We note that after each step of transformation, F(S,1) is decreased. We shall give this kind of transformation a special name.

Definition:

Let $S=(a_1,a_2,\ldots,a_N)$, $T=(a_1',a_2',\ldots,a_N')$ and $\prod_{i=1}^{N} a_i=\prod_{i=1}^{N} a_i'$. If in S, there exists an i such that $a_i=pq$ and in T, there exists a j such that $a_j'=pa_j$, $a_i'=q$ and for all k, $k\neq i,j$, $a_k'=a_k$, T is a pq-transformation of S.

Example 4.3

Let $S=(1,2,16)$ and $T=(2,2,8)$. T is a pq-transformation of S. In this case, $p=2,q=8,i=3,j=1$.

Definition:

Let T be a pq-transformation of S. If $F(T,1)<F(S,1)$, T is a successful pq-transformation of S.

Lemma 4.1

Let $S=(a_1,a_2,\ldots,a_N)$ and $T=(a_1',a_2',\ldots,a_N')$.

Let T be a pq-transformation of S. If in this pq-transformation, $p>1$ and $q>a_j$, T is a successful pq-transformation of S.

Proof:

Since T is a pq-transformation of S, there exists an i and a j such that in S, $a_i=pq$ and in T, $a_i'=q$ and $a_j'=pa_j$. Therefore,

$$\begin{aligned} & a_i+a_j-a_i'-a_j' \\ &= pq+a_j-q-pa_j \\ &= (p-1)(q-a_j)>0 \end{aligned}$$

Consequently,

$$\begin{aligned} & a_1'+a_2'+\ldots+a_i'+\ldots+a_j'+\ldots+a_N' \\ &< a_1+a_2+\ldots+a_i+\ldots+a_j+\ldots+a_N \end{aligned}$$

Hence T is a successful pq-transformation of S. Q.E.D.

Example 4.4

Let $S=(2,6,8)$. Since $a_3=8=2\times4$, we may choose $p=2$ and $q=4$. Applying this pq-transformation to S by letting j be 1, we obtain $T=(2\times2,6,4)=(4,6,4)$. It is easy to see that this is a successful pq-transformation.

Using Algorithm A and Lemma 4.1, we may prove the following:

Lemma 4.2

Let $S=(a_1,a_2,\ldots,a_N)$ where $\prod_{i=1}^{N} a_i=C$. S can be converted into a minimal N-tuple of C by finite number of successful pq-transformations.

Proof:

Note that in Algorithm A, the algorithm always terminates and produces an N-tuple in which every pair (a_i,a_j) is a minimal 2-tuple. Since, according to Lemma 4.1, every transform executed in the algorithm is a successful pq-transformation, we have the proof. Q.E.D.

In the following, we shall prove that after a successful pq-transformation, not only is F(S,1) reduced (by definition), F(S,2), ..., F(S,N-1) and all simultaneously reduced. Let us now first demonstrate this first by considering the case where N=4 and K=2.

Example 4.5

Let $S=(a,b,c,d)$, $T=(a,b',c',d)$ and T be a successful pq-transformation of S. In this example, we shall show that $F(T,2)<F(S,2)$. To show this, let us first note that

$$F(S,1)-F(T,1)$$
$$= a+b+c+d-(a+b'+c'+d)$$
$$= b+c-b'-c'>0$$

$$F(S,2)-F(T,2)$$
$$= (ab+ac+ad+bc+bd+cd)-(ab'+ac'+ad+b'c'+b'd+c'd)$$
$$= ((a+c+d)b+(a+d)c)-((a+b'+d)c'+(a+d)b')$$
$$= (b(a+c+d)-c'(a+b'+d))+(c-b')(a+d) \qquad (1)$$

Since T is a successful pq-transformation of S, we have $bc=b'c'$. Substituting this into (1), we have

$$F(S,2)-F(T,2)$$
$$= b(a+d)-c'(a+c)+(c-b')(a+d)$$
$$= (a+d)(b+c-b'-c')>0$$

We now prove the following lemma.

<u>Lemma 4.3</u>

Let $S=(a_1,a_2,\dots,a_N)$, $T=(a_1',a_2',\dots,a_N')$ and T be a successful pq-transformation of S. Then $F(T,K)<F(S,K)$, for $K=1,2,\dots,N-1$.

<u>Proof:</u>

Since T is a successful pq-transformation of S, we have

$$a_1+a_2+\dots+a_{i-1}+q+a_{i+1}+\dots+a_{j-1}+pa_j+a_{j+1}+\dots+a_N$$
$$< a_1+a_2+\dots+a_{i-1}+pq+a_{i+1}+\dots+a_{j-1}+a_j+a_{j+1}+\dots+a_N$$

Therefore,

$$q+pa_j<pq+a_j$$
$$(q-a_j)(p-1)>0$$
$$p>1 \text{ and } q>a_j$$
or
$$p<1 \text{ and } q<a_j.$$

Consider $F(S,K)-F(T,K)$.

$$F(S,K)-(F(T,K).$$

$$= \sum_{\substack{i_1=1\\ i_m\neq j\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}\dots a_{i_{K-1}}(a_j)+\sum_{\substack{i_1=1\\ i_m\neq i\\ i_m\neq j\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}\dots a_{i_{K-1}}(pq)$$

$$-\sum_{\substack{i_1=1\\ i_m\neq i\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}'a_{i_2}'\dots a_{i_{K-1}}'(q)-\sum_{\substack{i_1=1\\ i_m\neq j\\ i_m\neq i\\ i_1<i_2\ \dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}\dots a_{i_{K-1}}(pa_j)$$

$$= (a_j\sum_{\substack{i_1=1\\ i_m\neq j\\ i_m\neq i\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}\dots a_{i_{K-1}}-q\sum_{\substack{i_1=1\\ i_m\neq i\\ i_m\neq j\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}'a_{i_2}'\dots a_{K-1}')$$

$$+p(q-a_j)\sum_{\substack{i_1=1\\ i_m\neq j\\ i_m\neq i\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}\dots a_{i_{K-1}}$$

$$= (a_j\sum_{\substack{i_1=1\\ i_m\neq j\\ i_m\neq i\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}\dots a_{i_{K-1}}-q\sum_{\substack{i_1=1\\ i_m\neq i\\ i_m\neq j\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}\dots a_{i_{K-1}})$$

$$+p(q-a_j)\sum_{\substack{i_1=1\\ i_m\neq j\\ i_m\neq i\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}\dots a_{i_{K-1}}$$

$$= (a_j-q)\sum_{\substack{i_1=1\\ i_m\neq j\\ i_m\neq i\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}..a_{i_{K-1}}+p(q-a_j)\sum_{\substack{i_1=1\\ i_m\neq j\\ i_m\neq i\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}..a_{i_{K-1}}$$

$$= (q-a_j)(p-1)\sum_{\substack{i_1=1\\ i_m\neq i\\ i_m\neq j\\ i_1<i_2<\dots<i_{K-1}}}^{N} a_{i_1}a_{i_2}\dots a_{i_{K-1}}>0$$

Hence, $F(S,K)>F(T,K)$. Q.E.D.

Using Lemma 4.2 and Lemma 4.3, we can prove the following theorem.

<u>Theorem 4.1</u>

Let $S=(a_1,a_2,\dots,a_N)$ and $\prod_{i=1}^{N} a_i=C$. If S is not a minimal N-tuple of C, S can be transformed into $S'=(a_1',a_2',\dots,a_N')$ such that S' is a minimal N-tuple of C and $F(S',K)<F(S,K)$ for $K=1,2,\dots,N-1$.

<u>Proof:</u>

According to Lemma 4.2 and Algorithm A, we can apply a sequence of pq-transformations to S to transform S into S' such that S' is a minimal N-tuple of C. Assume that algorithm A takes M steps to finish. Let $S_0=S$ and after the exection of the m-th step, the N-tuple becomes S_m. We now have S_0, S_1, ..., S_M where $S_0=S$ and $S_M=S'$. According to Lemma 4.3, $F(S_i,K)<F(S_{i-1},K)$ for $i=1,2,\dots,M$ and $K=1,2,\dots,N-1$. In particular, $F(S',K)=F(S_M,K)<$

$F(S_0,K)=F(S,K)$. Thus the proof. Q.E.D.

Theorem 4.1 states that for any given N-tuple S of a constant C, if S is not minimal, we can always transform it into a minimal N-tuple of C. After this transformation, F(S,K) is reduced for all K=1,2,...,N-1.

Corollary 4.1:

If there is only one minimal N-tuple of a constant C, $\sum_{K=1}^{N-1} F(S,K)$ is the smallest among all possible N-tuples of C, iff S is the only minimal N-tuple of C.

Unfortunately, while there is only one minimal N-tuple for most cases, there are counter examples. We conducted a computerized checking through all integers from 1 to 1000 for N=3. We found that among these one thousand integers, integer 360 has two minimal 3-tuples, namely $S_1=(6,6,10)$ and $S_2=(5,8,9)$. It is interesting to note that this is the only counter example found among these one thousand integers. Furthermore, it should be noted that

$$F(S_1,1)=F(S_2,1)$$

$$F(S_1,2)=6\times6+6\times10+6\times10=156$$

and $F(S_2,2)=5\times8+5\times9+8\times9=157$.

Although $F(S_1,2)\neq F(S,2)$, the difference between them is small.

Let us conclude this section by the following statements:

(1) Given a constant C and an N-tuple $S=(a_1,a_2,\ldots,a_N)$ of C, if S is not a minimal N-tuple of C, S can be transformed into S' such that S' is a minimal N-tuple of C and $\sum_{K=1}^{N-1} F(S',K) < \sum_{K=1}^{N-1} F(S,K)$.

(2) For most constant C's, since there is only one minimal N-tuple of C, this minimal N-tuple S of C has the property that F(S',K) is minimized over all possible N-tuples of C.

Section 5. The Application of N-tuple Theories to the Design of Cartesian Product Files

In Section 3, we showed that the problem of designing an optimal Cartesian product file can be reduced to the problem of dividing each domain D_i into m_i subsets where each subset contains z_i elements. The values of $z_1,z_2,\ldots,z_N$ should staisfy the following conditions:

1. $z_1z_2\ldots z_N$=C=bucket size
2. $d_i/z_i=m_i$=an integer
3. $z_1+z_2+\ldots+z_N$
$+ z_1z_2+z_1z_3+\ldots+z_{N-1}z_N$
$+ \ldots$
$+ z_1z_2\ldots z_{N-1}+\ldots+z_2z_3\ldots z_N$ is minimized.

Using Theorem 4.1, we can obtain the following theorem.

Theorem 5.1

Let there be $NR=d_1d_2\ldots d_N$ records where d_i is the size of the domain D_i of attribute A_i. Let C be the bucket size. A Cartesian product file F is an optimal Cartesian product file if the records of each bucket are of the form of

$$D_{1s}\times D_{2s}\times\ldots\times D_{Ns}$$

where the size of D_{is} is z_i and z_i's satisfy the following conditions:

(1) $z_1z_2\ldots z_N=C$,

(2) $d_i/z_i=m_i$=an integer,

(3) $(z_1,z_2,\ldots,z_N)$ is the only minimal N-tuple of C.

To obtain a set of z_i's satisfying conditions (1) and (3), we may simply apply Algorithm A to the N-tuple (1,1,..., C). If we can rearrange the resulting N-tuple to be $(z_1,z_2,\ldots,z_N)$ in such a way that $d_i/z_i=m_i$=an integer for all $1\le i\le N$, and we are further sure that $(z_1,z_2,\ldots,z_N)$ is the only minimal N-tuple, then we have obtained an optimal Cartesian product file. Here, the following should be pointed out.

(1) It is very rare, as our experimental results demonstrate, that there is more than one minimal N-tuple for a constant C.

(2) Even if $S=(z_1,z_2,\ldots,z_N)$ is not the only minimal N-tuple, indicating that there might exist an $S'=(z_1,z_2,\ldots,z_N)$ such that F(S',K)<F(S,K) for some K, F(S',K) will not be significantly smaller than F(S,K). Besides, if such a K exists, this file is still optimal for queries with less than K queries specified.

Example 5.1

Let $d_1=8$, $d_2=4$, $d_3=6$ and C=32. Applying Algorithm A to (1,1,32), we obtain (2,4,4) as a minimal 3-tuple. It is not difficult to see that this is the only minimal 3-tuple of 32. We rearrange (2,4,4) into (4,4,2).

Then $d_1/z_1=8/4=2$
$d_2/z_2=4/4=1$
and $d_3/z_3=6/2=3$.

This means that D_1 should be divided into two subsets, D_2 into 1 subset and D_3 into three subsets. The resulting Cartesian Product file is an optimal Cartesian

product file.

If $d_1=d_2=\ldots=d_N$ in the resulting N-tuple S after Algorithm A is applied, $z_1=z_2=\ldots=z_N$. In this case, S is the only minimal N-tuple of C. If d_i/z_i=an integer for all i, the resulting Cartesian product file must be the optimum Cartesian product file for this set of records. This coincides with the result obtained by Rivest [1976].

Section 6. The Theories of Minimal N-tuple and Partial Match Patterns

In the previous sections, we assumed that all records in $D_1 \times D_2 \times \ldots \times D_N$ were present. This is obviously not a practical assumption. Unfortunately, it is difficult to design an optimal Cartesian product file for partial match queries where some records are missing. In this section, we shall introduce the multi-key hashing method [Rothnie and Lozano 1974] which does not require the assumption that all records are present. We thus introduce the partial match pattern concept defined by Lin, Lee and Du [1979]. Finally, we show how the theories of minimal N-tuples can be applied to design a Cartesian product file which is optimal with respect to partial match patterns.

The multi-key hashing method can be briefly defined as follows:

(1) Choose a hashing function g_i for domain D_i such that g_i: $D_i \rightarrow \{0,1,\ldots,m_i-1\}$ where $m_1 m_2 \ldots m_N$=NB, the total number of buckets required by the file.
(2) Associate with each N-tuple $(L_1,L_2,\ldots,L_N)$ a bucket where L_i is an integer, $0 \le L_i \le m_i-1$.
(3) If the attributes $A_1,A_2,\ldots,A_N$ of a record R have values $r_1,r_2,\ldots,r_N$ respectively, assign R into the bucket associated with $(g_1(r_1),g_2(r_2),\ldots,g_N(r_N))$ where $r_i \in D_i$ for $i=1,2,\ldots,N$.

Let us consider the case where $D_1=\{a,b,c,d\}$ and $D_2=\{e,f,g\}$. We can define the following hashing functions:

$$g_1(x)=0 \text{ if } x=a,b$$
$$\quad =1 \text{ if } x=c,d.$$

$$g_2(x)=0 \text{ if } x=e,f$$
$$\quad =1 \text{ if } x=g.$$

In this case, records will be hashed into their respective buckets as shown in Table 6.1. The reader should note that

---
Table 6.1 here
---

not all records are present. It should also be obvious that a file produced by the multi-key hashing method is a Cartesian product file.

If we ignore the overflow problem, the retrieval of the record $(r_1,r_2,\ldots,r_N)$ (every attribute is used.) needs examineing exactly one bucket. However, the partial match query with $A_i=r_i$ (for any r_i) examines NB/m_i buckets, the query with $A_i=r_i$ and $A_j=r_j$ (for any r_i and r_j) examines NB/$(m_i \cdot m_j)$ buckets, etc.

Lin, Lee and Du [1979] defined a partial match pattern to be a class of partial match queries.

If the partial match queries involve the same set of attributes, they belong to the same partial match pattern. For instance, the partial match query $A_i=r_i$ and $A_j=r_j$ belongs to the partial match pattern (A_i,A_j). The partial match query $A_i=s_i$ and $A_j=s_j$ belongs to the same partial match pattern (A_i,A_j).

Let us consider the case when N=3. The total number of buckets to be examined, over all possible partial match patterns involving exactly one attribute, is

$$\frac{NB}{m_1} + \frac{NB}{m_2} + \frac{NB}{m_3}$$

$$= NB\left(\frac{m_2m_3+m_1m_3+m_1m_2}{m_1m_2m_3}\right)$$

$$= m_1m_2 + m_1m_3 + m_2m_3 \quad (m_1m_2m_3=NB)$$

Similarly, the total number of buckets to be examined, over all possible partial match patterns involving exactly two attributes, is

$$\frac{NB}{m_1m_2} + \frac{NB}{m_1m_3} + \frac{NB}{m_2m_3}$$

$$= NB\,\frac{(m_3+m_2+m_1)}{m_1m_2m_3}$$

$$= m_1 + m_2 + m_3$$

The average number of buckets to be examined, over all possible partial match patterns, is

$$(m_1m_2+m_1m_3+m_2m_3+m_1+m_2+m_3)/(N+\binom{N}{2})$$

where $N+\binom{N}{2}$ is the total number of possible partial match patterns. Since this is a constant, to minimize the average number of buckets to be examined, we merely have to minimize

$$m_1+m_2+m_3+m_1m_2+m_1m_3+m_2m_3$$

under the constraint that $m_1m_2m_3$=NB.

In general, our problem of designing an optimal Cartesian product file for partial match patterns is as follows: Given NB, the total number of buckets and N, the total number of attributes, we should find an N-tuple $S=(m_1,m_2,\ldots,m_N)$ satisfying the following conditions:

(1) $m_1,m_2,\ldots,m_N$ are all integers.

(2) $\prod_{i=1}^{N} m_i = NB$.

(3) $\prod_{K=1}^{N-1} F(S,K)$ is minimized over all possible N-tuples satisfying (1) and (2).

The reader can now see that the theories developed in Section 5 are directly applicable to the partial match pattern problem. In fact, we can easily prove the following theorem.

Theorem 6.1

For the multi-key hashing method, if each record is characterized by N attributes and NB is the total number of buckets required in the file, then the average number of buckets examined, over all possible partial match patterns, is minimized when the hashing function divides each D_i into m_i subsets and the N-tuple $S=(m_1,m_2,\ldots,m_N)$ is the only minimal N-tuple of NB.

It should be noted that an optimal N-tuple $S=(m_1,m_2,\ldots,m_N)$ can be obtained by applying Algorithm A to $(1,1,\ldots,NB)$. If S is the only minimal N-tuple of NB, we have obtained an optimal solution. Since we expect most minimal N-tuples to be unique, we believe that Theorem 6.1 is very useful for constructing optimal files for partial match patterns. Even if a minimal N-tuple is not the only one, we still expect it to produce a file structure which is very close to an optimal one.

Finally, let us note that if $(NB)^{1/N}$ is an integer, there is only one minimal N-tuple of NB, namely, the N-tuple $(m_1, m_2,\ldots,m_N)$ where

$$m_1=m_2=\ldots=m_N=(NB)^{1/N}$$

In this case, we have got an optimal file for partial match patterns. This coincides with the result obtained by Lin, Lee and Du [1979].

We should emphasize here again that the multi-key hashing method does not require the assumption that all records have to be present, yet it still produces Cartesian product files. We can not guarantee that our method creates a file which is optimal with respect to partial match queries. We, however, can guarantee that our method is optimal with respect to partial match patterns.

Section 7. Future Research

Although we have made some progress in our research, we must admit that our results are still not practical because in practice, we may not be able to factor C. An extreme case is that C might be a prime number. Even if we successfully find z_1, z_2 ... and z_N, they may not satisfy the condition that d_i/z_i=an integer. One possible solution is that we find z_i's such that

$$z_1 z_2 \ldots z_N \doteq C$$

and $d_i/z_i \geq 1$ for all i.

REFERENCES

1. Bently, J. L. (1979): Multidimensional Binary Search Trees in Database Applications, IEEE Trans. on Software Engineering, Vol. SE-5, No. 4, July 1979, pp.333-342.

2. Lee, R. C. T. and Tseng, S. H. (1979): Multi-key Sorting, Policy Analysis and Information Systems, Vol. 3, No. 2, Dec. 1979, pp.1-20.

3. Lin, W. C., Lee, R. C. T. and Du. H. C. (1979); Common Properties of Some Multi-attribute File Systems, IEEE Trans. on Software Engineering, Vol. SE-5, No. 2, March 1979, pp.160-174.

4. Liou, J. H. and Yao, S. B. (1977): Multi-dimensional Clustering for Data Base Organizations, Information Systems, Vol. 2, 1977, pp.187-198.

5. Rivest, R. L. (1976): Partial-match Retrieval Algorithms, SIAM J. Comput., Vol. 15, No. 1, March 1976. pp.19-50.

6. Rothnie, J. B. and Lozano, T. (1974): Attribute Based File Organization in a Paged Memory Environment, Comm. ACM., Vol. 17, No. 2, Feb. 1974, pp.63-69.

7. Slagle, J. R., Chang, C. L. and Lee, R. C. T. (1974): Experiments with Some Clustering Analysis Algorithm, Pattern Recognition, Vol. 6, 1974.

Bucket 1	Bucket 2	Bucket 3	Bucket 4
(a,a)	(a,b)	(a,c)	(a,d)
(b,b)	(b,c)	(b,c)	(b,a)
(c,c)	(c,d)	(c,a)	(c,b)
(d,d)	(d,a)	(d,b)	(d,c)

Table 1.1(a)

Queries	Buckets to be examined
(a,*)	1, 2, 3, 4
(b,*)	1, 2, 3, 4
(c,*)	1, 2, 3, 4
(d,*)	1, 2, 3, 4
(*,a)	1, 2, 3, 4
(*,b)	1, 2, 3, 4
(*,c)	1, 2, 3, 4
(*,d)	1, 2, 3, 4

Table 1.1(b)

Bucket 1	Bucket 2	Bucket 3	Bucket 4
(a,a)	(a,c)	(c,a)	(c,c)
(a,b)	(a,d)	(c,b)	(c,d)
(b,a)	(b,c)	(d,a)	(d,c)
(b,b)	(b,d)	(d,b)	(d,d)

Table 1.2(a)

Queries	Buckets to be examined
(a,*)	1, 2
(b,*)	1, 2
(c,*)	3, 4
(d,*)	3, 4
(*,a)	1, 3
(*,b)	1, 3
(*,c)	2, 4
(*,d)	2, 4

Table 1.2(b)

Data Set	d_1	d_2	d_3	d_4	NR	NB	BZ	ANB_R	ANB_{CP}	ANB_{CP}/ANB_R
1	3	4	5	-	60	4	15	2.9153	1.1695	0.7441
2	4	2	3	-	24	4	6	2.6857	1.9429	0.7234
3	2	5	5	-	50	5	10	3.0526	2.1930	0.7185
4	1	4	4	-	16	4	4	2.0000	1.5758	0.7879
5	4	5	5	-	100	10	10	4.9241	3.1646	0.6427
6	3	3	2	-	18	3	6	2.2414	1.7586	0.7846
7	4	3	5	-	60	6	10	3.6271	2.5424	0.7009
8	3	5	2	-	30	3	10	2.2439	1.8293	0.8152
9	5	5	3	-	75	5	15	3.1765	2.3519	0.7080
10	3	4	5	5	300	10	30	4.9714	2.6969	0.5425
11	2	3	5	4	120	6	20	3.4728	2.1841	0.6289
12	3	2	2	3	36	4	9	2.6355	2.0187	0.7660
13	4	2	4	4	128	8	16	3.9512	2.0813	0.5267
14	5	5	2	2	100	10	10	3.8969	2.7345	0.7020
15	4	3	3	5	180	10	18	4.6856	2.5753	0.5496
16	5	2	3	3	90	3	30	2.3655	1.7208	0.7275
17	4	3	4	5	240	10	24	4.8273	2.6462	0.5482
18	3	1	2	5	30	3	10	1.8443	1.5000	0.8133
19	4	3	5	2	120	10	12	4.2197	2.4686	0.5978
20	2	2	2	4	32	4	8	2.6078	1.7647	0.6767
21	2	3	3	4	72	3	24	2.3892	2.0299	0.8496
22	5	5	3	3	225	9	25	4.6047	3.0343	0.6588
23	2	3	-	-	6	3	2	2.0000	1.8000	0.9000
24	2	4	-	-	8	2	4	1.6667	1.3333	0.8000
25	2	5	-	-	10	5	2	2.5714	2.1429	0.8333
26	4	3	-	-	12	3	4	2.4286	2.1429	0.8824
27	4	4	-	-	16	4	4	3.1250	2.0000	0.6400
28	5	3	-	-	15	5	3	3.1250	2.5000	0.8000
29	5	5	-	-	25	5	5	3.8000	3.0000	0.7895
30	4	5	-	-	20	2	10	2.0000	1.5556	0.7778

Table 2.1

d_i: the size of domain D_i of attribute A_i.
NR: the total number of records.
NB: the total number of buckets.
BZ: the block size.
ANR_R: the average number of buckets accessed per partial match query for random files.
ANB_{CP}: the average number of buckets accessed if a near-optimal Cartesian product file is used.

Attribute 1	Attribute 2	N-tuple (N=2)	Bucket Number
(a	e)	(0, 0)	1
(a	f)		
(b	e)		
(b	f)		
(a	g)	(0, 1)	2
(b	g)		
(c	e)	(1, 0)	3
(c	f)		
(d	e)		
(d	f)		
(c	g)	(1, 1)	4
(d	g)		

Table 6.1

International Conference on Management of Data archive

Proceedings of the 1980 ACM SIGMOD international conference on Management of data >citation

1980

Table of Contents